# CS 473: Algorithms

Chandra Chekuri
chekuri@cs.illinois.edu
3228 Siebel Center

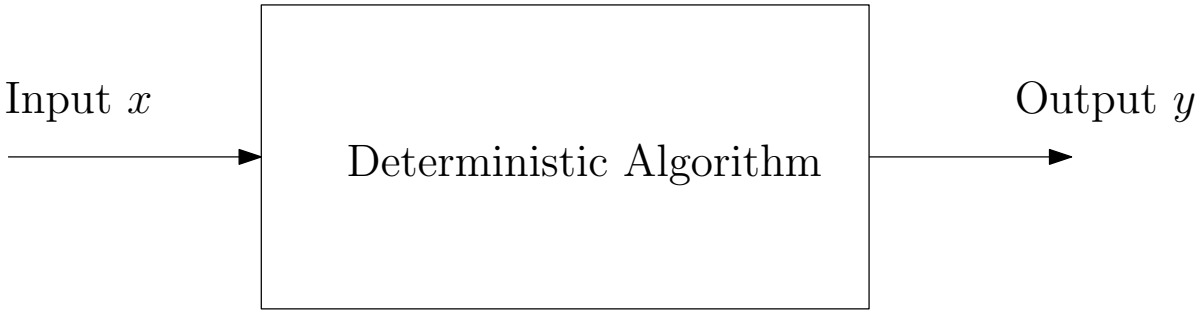
University of Illinois, Urbana-Champaign

Fall 2010

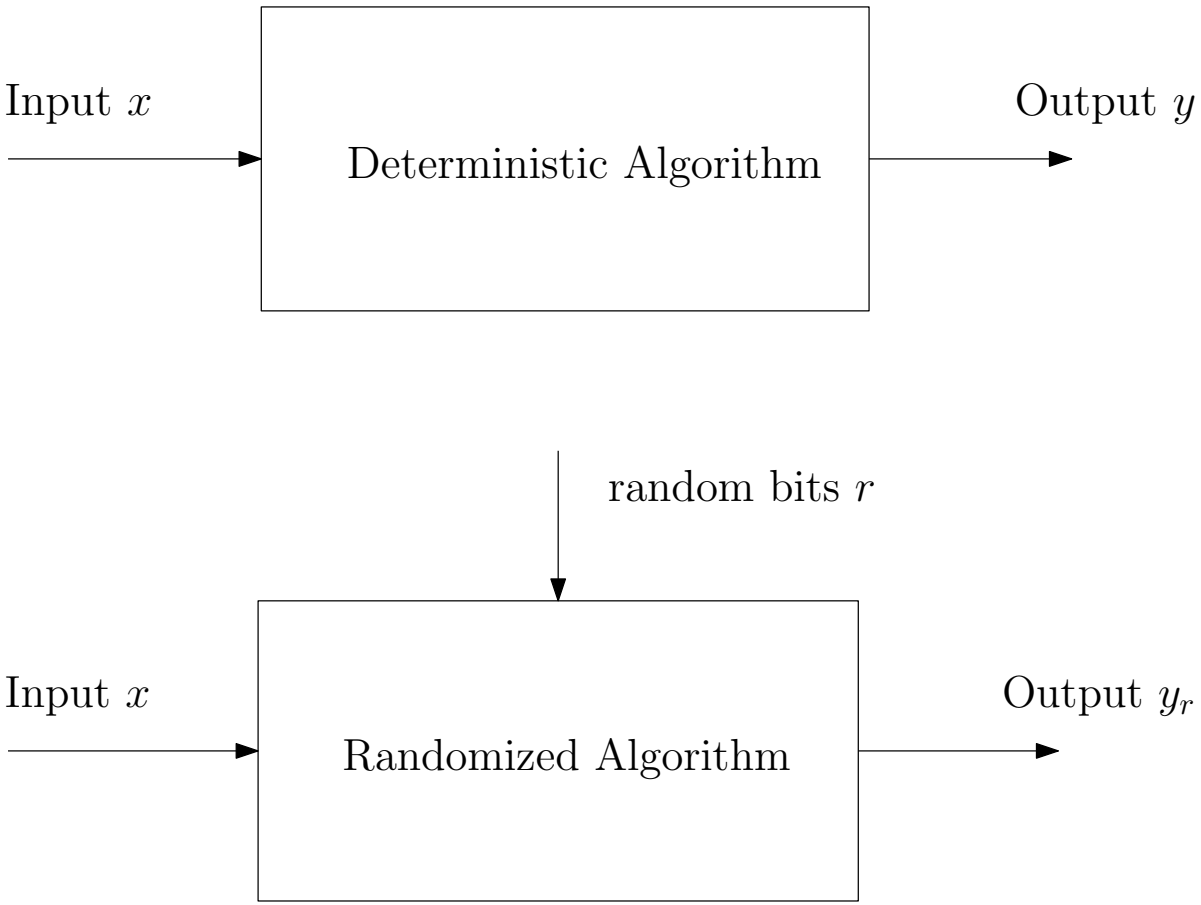
Part I

# Introduction to Randomized Algorithms

# Randomized Algorithms



# Randomized Algorithms



# Example: Randomized Quicksort

## Quick Sort[Hoare]

1. Pick a pivot element from array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
3. Recursively sort the subarrays, and concatenate them.

## Randomized Quick Sort

1. Pick a pivot element *uniformly at random* from the array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
3. Recursively sort the subarrays, and concatenate them.

# Example: Randomized Quicksort

Recall: Quick Sort can take $\Omega(n^2)$ time to sort array of size $n$.

# Example: Randomized Quicksort

Recall: Quick Sort can take $\Omega(n^2)$ time to sort array of size $n$.

**Theorem**

*Randomized Quick Sort sorts a given array of length n in* $O(n \log n)$ expected *time.*

# Example: Randomized Quicksort

Recall: Quick Sort can take $\Omega(n^2)$ time to sort array of size $n$.

**Theorem**

*Randomized Quick Sort sorts a given array of length n in $O(n \log n)$ expected time.*

**Note:** On *every* input randomized quick sort takes $O(n \log n)$ time in expectation. On *every* input it may take $\Omega(n^2)$ time with some small probability.

# Example: Verifying Matrix Multiplication

## Problem

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

# Example: Verifying Matrix Multiplication

**Problem**

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Deterministic algorithm:

- Multiply $A$ and $B$ and check if equal to $C$.
- Running time?

# Example: Verifying Matrix Multiplication

**Problem**

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Deterministic algorithm:

- Multiply $A$ and $B$ and check if equal to $C$.
- Running time? $O(n^3)$ by straight forward approach. $O(n^{2.37})$ with fast matrix multiplication (complicated and impractical).

# Example: Verifying Matrix Multiplication

## Problem

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

# Example: Verifying Matrix Multiplication

## Problem

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Randomized algorithm:

- Pick a random $n \times 1$ vector $r$.
- Return the answer of the equality $ABr = Cr$.
- Running time?

# Example: Verifying Matrix Multiplication

## Problem

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Randomized algorithm:

- Pick a random $n \times 1$ vector $r$.
- Return the answer of the equality $ABr = Cr$.
- Running time? $O(n^2)$!

# Example: Verifying Matrix Multiplication

## Problem

Given three $n \times n$ matrices $A, B, C$ is $AB = C$?

Randomized algorithm:

- Pick a random $n \times 1$ vector $r$.
- Return the answer of the equality $ABr = Cr$.
- Running time? $O(n^2)$!

## Theorem

*If $AB = C$ then the algorithm will always say YES. If $AB \neq C$ then the algorithm will say YES with probability at most $1/2$. Can repeat the algorithm 100 times independently to reduce the probability of a false positive to $1/2^{100}$.*

# Why randomized algorithms?

- Many many applications in algorithms, data structures and computer science!
- In some cases only known algorithms are randomized or randomness is provably necessary.
- Often randomized algorithms are (much) simpler and/or more efficient.
- Several deep connections to mathematics, physics etc.
- ...
- Lots of fun!

# Where do I get random bits?

**Question:** Are true random bits available in practice?

- Can use pseudo-random bits or semi-random bits from nature. Several fundamental unresolved questions in complexity theory on this topic. Beyong the scope of this course.
- In practice pseudo-random generators work quite well in many applications.
- The model is interesting to think in the abstract and is very useful even as a theoretical construct. One can *derandomize* randomized algorithms to obtain deterministic algorithms.

# Average case analysis vs Randomized algorithms

**Average case analysis:**

- Fix a deterministic algorithm.
- Assume inputs comes from a probability distribution.
- Analyze the algorithm's *average* performance over the distribution over inputs.

**Randomized algorithms:**

- Algorithm uses random bits in addition to input.
- Analyze algorithms *average* performance over the given input where the average is over the random bits that the algorithm uses.
- On each input behaviour of algorithm is random. Analyze worst-case over all inputs of the (average) performance.

# Discrete Probability

We restrict attention to finite probability spaces.

## Definition

A discrete probability space is a pair $(\Omega, p)$ consists of finite set $\Omega$ of *elementary* events and function $p : \Omega \rightarrow [0, 1]$ which assigns a probability $p(\omega)$ for each $\omega \in \Omega$ such that $\sum_{\omega \in \Omega} p(\omega) = 1$.

# Discrete Probability

We restrict attention to finite probability spaces.

## Definition

A discrete probability space is a pair $(\Omega, p)$ consists of finite set $\Omega$ of *elementary* events and function $p : \Omega \rightarrow [0, 1]$ which assigns a probability $p(\omega)$ for each $\omega \in \Omega$ such that $\sum_{\omega \in \Omega} p(\omega) = 1$.

## Example

An ubiased coin. $\Omega = \{H, T\}$ and $p(H) = p(T) = 1/2$.

## Example

A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(i) = 1/6$ for $1 \leq i \leq 6$.

## Example

A biased coin. $\Omega = \{H, T\}$ and $p(H) = 2/3, p(T) = 1/3$

# More Examples

## Example

Two independent ubiased coins. $\Omega = \{HH, TT, HT, TH\}$ and $p(HH) = p(TT) = p(HT) = p(TH) = 1/4$.

## Example

A pair of (highly) correlated dice.
$\Omega = \{(i,j) \mid 1 \leq i \leq 6, 1 \leq j \leq 6\}$.
$p(i,i) = 1/6$ for $1 \leq i \leq 6$ and $p(i,j) = 0$ if $i \neq j$.

# Events

## Definition

Given a probability space $(\Omega, p)$ an *event* is a subset of $\Omega$. In other words an event is a collection of elementary events. The probability of an event $A$, denoted by $p(A)$, is $\sum_{\omega \in A} p(\omega)$. The complement of an event $A \subseteq \Omega$ is the event $\Omega \setminus A$ frequently denoted by $\bar{A}$.

# Events

## Definition

Given a probability space $(\Omega, p)$ an *event* is a subset of $\Omega$. In other words an event is a collection of elementary events. The probability of an event $A$, denoted by $p(A)$, is $\sum_{\omega \in A} p(\omega)$. The complement of an event $A \subseteq \Omega$ is the event $\Omega \setminus A$ frequently denoted by $\bar{A}$.

## Example

A pair of independent dice. $\Omega = \{(i,j) \mid 1 \leq i \leq 6, 1 \leq j \leq 6\}$.

- Let $A$ be the event that the sum of the two numbers on the dice is even. Then $A = \{(i,j) \in \Omega \mid (i+j) \text{ is even}\}$.
  $p(A) = |A|/36 = 1/2$.
- Let $B$ be the event that the first die has 1. Then
  $B = \{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6)\}$.
  $p(B) = 6/36 = 1/6$.

# Independent Events

## Definition

Given a probability space $(\Omega, p)$ and two events $A, B$ are *independent* if and only if $p(A \cap B) = p(A)p(B)$. Otherwise they are *dependent*. In other words $A, B$ independent implies one does not affect the other.

# Independent Events

## Definition

Given a probability space $(\Omega, p)$ and two events $A, B$ are *independent* if and only if $p(A \cap B) = p(A)p(B)$. Otherwise they are *dependent*. In other words $A, B$ independent implies one does not affect the other.

## Example

Two coins. $\Omega = \{HH, TT, HT, TH\}$ and $p(HH) = p(TT) = p(HT) = p(TH) = 1/4$.

- $A$ is the event that the first coin is heads and $B$ is the event that second coin is tails. $A, B$ are independent.
- $A$ is the event that the two coins are different. $B$ is the event that the second coin is heads. $A, B$ independent.
- $A$ is the event that both are not tails and $B$ is event that

# Random Variables

## Definition

Given a probability space $(\Omega, p)$ a (real-valued) random variable $X$ over $\Omega$ is a function that maps each elementary event to a real number. In other words $X : \Omega \rightarrow \mathbb{R}$.

# Random Variables

## Definition

Given a probability space $(\Omega, p)$ a (real-valued) random variable $X$ over $\Omega$ is a function that maps each elementary event to a real number. In other words $X : \Omega \rightarrow \mathbb{R}$.

## Example

A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(i) = 1/6$ for $1 \leq i \leq 6$.

- $X : \Omega \rightarrow \mathbb{R}$ where $X(i) = i \mod 2$.
- $Y : \Omega \rightarrow \mathbb{R}$ where $Y(i) = i^2$.

# Random Variables

## Definition

Given a probability space $(\Omega, p)$ a (real-valued) random variable $X$ over $\Omega$ is a function that maps each elementary event to a real number. In other words $X : \Omega \rightarrow \mathbb{R}$.

## Example

A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(i) = 1/6$ for $1 \leq i \leq 6$.

- $X : \Omega \rightarrow \mathbb{R}$ where $X(i) = i \mod 2$.
- $Y : \Omega \rightarrow \mathbb{R}$ where $Y(i) = i^2$.

## Definition

A binary random variable is one that takes on values in $\{0, 1\}$.

# Indicator Random Variables

Special type of random variables that are quite useful.

## Definition

Given a probability space $(\Omega, p)$ and an event $A \subseteq \Omega$ the indicator random variable $X_A$ is a binary random variable where $X_A(\omega) = 1$ if $\omega \in A$ and $X_A(\omega) = 0$ if $\omega \notin A$.

# Indicator Random Variables

Special type of random variables that are quite useful.

## Definition

Given a probability space $(\Omega, p)$ and an event $A \subseteq \Omega$ the indicator random variable $X_A$ is a binary random variable where $X_A(\omega) = 1$ if $\omega \in A$ and $X_A(\omega) = 0$ if $\omega \notin A$.

## Example

A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(i) = 1/6$ for $1 \leq i \leq 6$. Let $A$ be the even that $i$ is divisible by 3. Then $X_A(i) = 1$ if $i = 3, 6$ and 0 otherwise.

# Expectation

## Definition

For a random variable $X$ over a probability space $(\Omega, p)$ the *expectation* of $X$ is defined as $\sum_{\omega \in \Omega} p(\omega)X(\omega)$. In other words the average value of $X$ according to the probabilities given by $p$.

# Expectation

## Definition

For a random variable $X$ over a probability space $(\Omega, p)$ the *expectation* of $X$ is defined as $\sum_{\omega \in \Omega} p(\omega)X(\omega)$. In other words the average value of $X$ according to the probabilities given by $p$.

## Example

A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(i) = 1/6$ for $1 \leq i \leq 6$.

- $X : \Omega \rightarrow \mathbb{R}$ where $X(i) = i \mod 2$. Then $E[X] = 1/2$.
- $Y : \Omega \rightarrow \mathbb{R}$ where $Y(i) = i^2$. Then $E[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = 91/6$.

# Expectation

## Definition

For a random variable $X$ over a probability space $(\Omega, p)$ the *expectation* of $X$ is defined as $\sum_{\omega \in \Omega} p(\omega)X(\omega)$. In other words the average value of $X$ according to the probabilities given by $p$.

## Example

A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $p(i) = 1/6$ for $1 \leq i \leq 6$.

- $X : \Omega \rightarrow \mathbb{R}$ where $X(i) = i \mod 2$. Then $E[X] = 1/2$.
- $Y : \Omega \rightarrow \mathbb{R}$ where $Y(i) = i^2$. Then $E[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = 91/6$.

## Proposition

*For an indicator variable $X_A$, $E[X_A] = p(A)$.*

# Linearity of Expectation

## Lemma

*Let $X, Y$ be two random variables over a probability space $(\Omega, p)$. Then $E[X + Y] = E[X] + E[Y]$.*

## Proof.

$$E[X + Y] = \sum_{\omega \in \Omega} p(\omega)(X(\omega) + Y(\omega))$$
$$= \sum_{\omega \in \Omega} p(\omega)X(\omega) + \sum_{\omega \in \Omega} p(\omega)Y(\omega) = E[X] + E[Y].$$

# Linearity of Expectation

## Lemma

*Let $X, Y$ be two random variables over a probability space $(\Omega, p)$. Then $E[X + Y] = E[X] + E[Y]$.*

## Proof.

$$E[X + Y] = \sum_{\omega \in \Omega} p(\omega)(X(\omega) + Y(\omega))$$
$$= \sum_{\omega \in \Omega} p(\omega)X(\omega) + \sum_{\omega \in \Omega} p(\omega)Y(\omega) = E[X] + E[Y].$$

□

## Corollary

$E[a_1X_1 + a_2X_2 + \ldots + a_nX_n] = \sum_{i=1}^{n} a_iE[X_i]$.

# Types of Randomized Algorithms

Typically one encounters the following types:

- *Las Vegas randomized algorithms:* for a given input $x$ output of algorithm is always correct but the running time is a random variable. In this case we are interested in analyzing the *expected* running time.

# Types of Randomized Algorithms

Typically one encounters the following types:

- *Las Vegas randomized algorithms:* for a given input $x$ output of algorithm is always correct but the running time is a random variable. In this case we are interested in analyzing the *expected* running time.
- *Monte Carlo randomized algorithms:* for a given input $x$ the running time is deterministic but the output is random; correct with some probability. In this case we are interested in analyzing the *probability* of the correct output (and also the running time).

# Types of Randomized Algorithms

Typically one encounters the following types:

- *Las Vegas randomized algorithms:* for a given input $x$ output of algorithm is always correct but the running time is a random variable. In this case we are interested in analyzing the *expected* running time.
- *Monte Carlo randomized algorithms:* for a given input $x$ the running time is deterministic but the output is random; correct with some probability. In this case we are interested in analyzing the *probability* of the correct output (and also the running time).
- Algorithms whose running time and output may both be random.

# Analyzing Las Vegas Algorithms

*Deterministic* algorithm $Q$ for a problem $\Pi$:

- Let $Q(x)$ be the time for $Q$ to run on input $x$ of length $|x|$.
- Worst-case analysis: run time on worst input for a given size $n$.

$$T_{wc}(n) = \max_{x:|x|=n} Q(x)$$

# Analyzing Las Vegas Algorithms

*Deterministic* algorithm $Q$ for a problem $\Pi$:

- Let $Q(x)$ be the time for $Q$ to run on input $x$ of length $|x|$.
- Worst-case analysis: run time on worst input for a given size $n$.

$$T_{wc}(n) = \max_{x:|x|=n} Q(x)$$

*Randomized* algorithm $Q$ for a problem $\Pi$:

- Let $Q(x)$ be the time for $Q$ to run on input $x$ of length $|x|$.
- $Q(x)$ is a random variable: depends on random bits used by $Q$
- $E[Q(x)]$ is the expected time for $Q$ on $x$
- Worst-case analysis: expected time on worst input of size $n$

$$T_{rand-wc}(n) = \max_{x:|x|=n} E[Q(x)]$$

# Analyzing Monte Carlo Algorithms

*Randomized* algorithm $Q$ for a problem $\Pi$:

- Let $Q(x)$ be the time for $Q$ to run on input $x$ of length $|x|$. For Monte Carlo, assumption is that run time is deterministic.
- Let $P(x)$ be the probability that $Q$ is correct on $x$
- $P(x)$ is a random variable: depends on random bits used by $Q$
- Worst-case analysis: success probability on worst input

$$P_{rand-wc}(n) = \min_{x:|x|=n} P(x)$$

Part II

# Randomized Quick Sort and Selection

# Randomized Quick Sort

## Randomized Quick Sort

1. Pick a pivot element *uniformly at random* from the array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
3. Recursively sort the subarrays, and concatenate them.

# Example

- array: 16, 12, 14, 20, 5, 3, 18, 19, 1

# Analysis via Recurrence

- Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized quick sort on $A$.
- Note that $Q(A)$ is a random variable
- Let $A^i_{\text{left}}$ and $A^i_{\text{right}}$ be the left and right arrarys obtained if pivot is rank $i$ element of $A$.

$$Q(A) = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

# Analysis via Recurrence

- Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized quick sort on $A$.
- Note that $Q(A)$ is a random variable
- Let $A^i_{\text{left}}$ and $A^i_{\text{right}}$ be the left and right arrarys obtained if pivot is rank $i$ element of $A$.

$$Q(A) = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

Since each element of $A$ has probably exactly of $1/n$ of being chosen:

$$Q(A) = n + \sum_{i=1}^{n} \frac{1}{n} \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

# Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} E[Q(A)]$ be the worst-case expected running time of randomized quick sort on arrays of size $n$.

# Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} E[Q(A)]$ be the worst-case expected running time of randomized quick sort on arrays of size $n$.

We have, for any $A$:

$$Q(A) = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

# Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} E[Q(A)]$ be the worst-case expected running time of randomized quick sort on arrays of size $n$.

We have, for any $A$:

$$Q(A) = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

Therefore, by linearity of expectation:

$$E[Q(A)] = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(E[Q(A^i_{\text{left}}])] + E[Q(A^i_{\text{right}})]\right)$$

# Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} E[Q(A)]$ be the worst-case expected running time of randomized quick sort on arrays of size $n$.

We have, for any $A$:

$$Q(A) = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

Therefore, by linearity of expectation:

$$E[Q(A)] = n + \sum_{i=1}^{n} \Pr[\text{pivot has rank } i] \left(E[Q(A^i_{\text{left}}]) + E[Q(A^i_{\text{right}})]\right)$$

$$\Rightarrow E[Q(A)] \leq n + \sum_{i=1}^{n} \frac{1}{n} \left(T(i-1) + T(n-i)\right)$$

# Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} E[Q(A)]$ be the worst-case expected running time of randomized quick sort on arrays of size $n$.

# Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} E[Q(A)]$ be the worst-case expected running time of randomized quick sort on arrays of size $n$. We derived:

$$E[Q(A)] \leq n + \sum_{i=1}^{n} \frac{1}{n} (T(i-1) + T(n-i))$$

Note that above holds for any $A$ of size $n$. Therefore

$$\max_{A:|A|=n} E[Q(A)] = T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n} (T(i-1) + T(n-i))$$

# Solving the Recurrence

$$T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n} \left( T(i-1) + T(n-i) \right)$$

with base case $T(1) = 0$.

# Solving the Recurrence

$$T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n} \left( T(i-1) + T(n-i) \right)$$

with base case $T(1) = 0$.

**Lemma**

$T(n) = O(n \log n)$.

# Solving the Recurrence

$$T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n} \left( T(i-1) + T(n-i) \right)$$

with base case $T(1) = 0$.

**Lemma**

$T(n) = O(n \log n)$.

**Proof.**

(Guess and) Verify by induction. □

# A Slick Analysis

Let $Q(A)$ be number of comparisons done on input array $A$:

- For $1 \leq i < j < n$ let $R_{ij}$ be the event that rank $i$ element is compared with rank $j$ element.
- $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise 0.

# A Slick Analysis

Let $Q(A)$ be number of comparisons done on input array $A$:

- For $1 \leq i < j < n$ let $R_{ij}$ be the event that rank $i$ element is compared with rank $j$ element.
- $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise 0.

$$Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$$

and hence by linearity of expecatation,

$$E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}]$$

# A Slick Analysis

**Question:** What is $\Pr[R_{ij}]$?

# A Slick Analysis

**Question:** What is Pr$[R_{ij}]$?

## Lemma

$Pr[R_{ij}] = \frac{2}{(j-i+1)}$.

# A Slick Analysis

**Question:** What is $\Pr[R_{ij}]$?

## Lemma

$Pr[R_{ij}] = \frac{2}{(j-i+1)}$.

## Proof.

Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be sort of $A$. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

**Observation:** If pivot is chosen outside $S$ then all of $S$ either in left array or right array.

**Observation:** $a_i$ and $a_j$ separated when a pivot is chosen from $S$ for the first time. Once separated no comparison.

**Observation:** $a_i$ is compared with $a_j$ if and only if either $a_i$ or $a_j$ is chosen as a pivot from $S$ at separation.

continued ....

# A Slick Analysis

## Lemma

$Pr[R_{ij}] = \frac{2}{(j-i+1)}$.

## Proof.

Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be sort of $A$. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

**Observation:** $a_i$ is compared with $a_j$ if and only if either $a_i$ or $a_j$ is chosen as a pivot from $S$ at separation.

**Observation:** Given that pivot is chosen from $S$ the probability that it is $a_i$ or $a_j$ is exactly $2/|S| = 2/(j-i+1)$ since the pivot is chosen uniformly at random from the array. □

# A Slick Analysis

$$E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}]$$

**Lemma**

$Pr[R_{ij}] = \frac{2}{(j-i+1)}$.

# A Slick Analysis

$$E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}]$$

**Lemma**

$Pr[R_{ij}] = \frac{2}{(j-i+1)}$.

$$\begin{aligned}
E[Q(A)] &= \sum_{1 \leq i < j \leq n} \Pr[R_{ij}] = \sum_{1 \leq i < j \leq n} \frac{2}{j-i+1} \\
&= \sum_{1 \leq i < n} \sum_{i < j \leq n} \frac{2}{j-i+1} = 2 \sum_{1 \leq i < n} \sum_{i < j \leq n} \frac{1}{j-i+1} \\
&= 2 \sum_{1 \leq i < n} (H_{n-i+1} - 1) \leq 2 \sum_{1 \leq i < n} H_n \\
&\leq 2nH_n = O(n \log n)
\end{aligned}$$

# Randomized Quick Selection

**Input** Unsorted array $A$ of $n$ integers

**Goal** Find the $j$'th smallest number in $A$ (*rank* $j$ number)

## Randomized Quick Selection

1. Pick a pivot element *uniformly at random* from the array
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
3. Return pivot if rank of pivot is $j$
4. Otherwise recurse on one of the arrays depending on $j$ and their sizes.

# Algorithm for Randomized Selection

**Assume** for simplicity that $A$ has distinct elements.

```
Random-SELECT(A, j):
    Pick pivot x uniformly at random from A
    Partition A into A_less, x, and A_greater using x as pivot
    If (|A_less|) = j − 1 return x
    Else if (|A_less|) ≥ j)
        return SELECT(A_less, j)
    Else
        return SELECT(A_greater, j − |A_less| − 1)
```

# Analysis via Recurrence

- Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized selection on $A$ for selecing rank $j$ lement.
- Note that $Q(A)$ is a random variable
- Let $A^i_{\text{less}}$ and $A^i_{\text{greater}}$ be the left and right arrarys obtained if pivot is rank $i$ element of $A$.
- Algorithm recurses on $A^i_{\text{less}}$ if $j < i$ and recurses on $A^i_{\text{greater}}$ if $j > i$ and terminates if $j = i$.

# Analysis via Recurrence

- Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized selection on $A$ for selecing rank $j$ lement.
- Note that $Q(A)$ is a random variable
- Let $A^i_{\text{less}}$ and $A^i_{\text{greater}}$ be the left and right arrarys obtained if pivot is rank $i$ element of $A$.
- Algorithm recurses on $A^i_{\text{less}}$ if $j < i$ and recurses on $A^i_{\text{greater}}$ if $j > i$ and terminates if $j = i$.

$$\begin{aligned} Q(A) &= n + \sum_{i=1}^{j-1} \Pr[\text{pivot has rank } i] Q(A^i_{\text{greater}}) \\ &\quad + \sum_{i=j+1}^{n} \Pr[\text{pivot has rank } i] Q(A^i_{\text{less}}) \end{aligned}$$

# Analyzing the Recurrence

As in quick sort we obtain the following recurrence where $T(n)$ is the worst-case expected time.

$$T(n) \leq n + \frac{1}{n}(\sum_{i=1}^{j-1} T(n-i) + \sum_{i=j}^{n} T(i-1))$$

**Theorem**

$T(n) = O(n)$.

**Proof.**

(Guess and) Verify by induction (see next slide). □

# Analyzing the recurrence

**Theorem**

$T(n) = O(n)$.

Prove by induction that $T(n) \leq \alpha n$ for some constant $\alpha \geq 1$ to be fixed later.

**Base case:** $n = 1$, we have $T(1) = 0$ since no comparisons needed and hence $T(1) \leq \alpha$.

**Induction step:** Assume $T(k) \leq \alpha k$ for $1 \leq k < n$ and prove it for $T(n)$. We have by the recurrence:

$$\begin{aligned} T(n) &\leq n + \frac{1}{n}(\sum_{i=1}^{j-1} T(n-i) + \sum_{i=j^n} T(i-1)) \\ &\leq n + \frac{\alpha}{n}(\sum_{i=1}^{j-1}(n-i) + \sum_{i=j}^{n}(i-1)) \quad \text{by applying induction} \end{aligned}$$

# Analyzing the recurrence

$$
\begin{aligned}
T(n) &\leq n + \frac{\alpha}{n}\left(\sum_{i=1}^{j-1}(n-i) + \sum_{i=j}^{n}(i-1)\right) \\
&\leq n + \frac{\alpha}{n}((j-1)(2n-j)/2 + (n-j+1)(n+j-2)/2) \\
&\leq n + \frac{\alpha}{2n}(n^2 + 2nj - 2j^2 - 3n + 4j - 2) \\
&\quad \text{above expression maximized when } j = (n+1)/2\text{: calculus} \\
&\leq n + \frac{\alpha}{2n}(3n^2/2 - n) \quad \text{substituting } (n+1)/2 \text{ for } j \\
&\leq n + 3\alpha n/4 \\
&\leq \alpha n \quad \text{for any constant } \alpha \geq 4
\end{aligned}
$$

# Comments on analyzing the recurrence

- Algebra looks messy but intuition suggest that the median is the hardest case and hence can plug $j = n/2$ to simplify without calculus
- Analyzing recurrences comes with practice and after a while one can see things more intuitively

**John Von Neumann**:
*Young man, in mathematics you don't understand things. You just get used to them.*